Executorch QNN Environments

Downlad and Unzip QNN SDK(v2.26)

```
cd /executorch
wget
https://softwarecenter.qualcomm.com/api/download/software/qualcomm_neural_processi
ng_sdk/v2.26.0.240828.zip
unzip v2.26.0.240828.zip -d /opt
```

Setup QNN

```
export ANDROID_NDK_ROOT=/opt/android-sdk/ndk/26.3.11579264
export QNN_SDK_ROOT=/opt/qairt/2.26.0.240828
export EXECUTORCH_ROOT=/executorch/
export LD_LIBRARY_PATH=$QNN_SDK_ROOT/lib/x86_64-linux-clang/:$LD_LIBRARY_PATH
export PYTHONPATH=$EXECUTORCH_ROOT/python:$PYTHONPATH
cd /executorch && cp schema/program.fbs exir/_serialize/program.fbs
cd /executorch && cp schema/scalar_type.fbs exir/_serialize/scalar_type.fbs
```

Llama Dependencies Install

```
source /llama/llama-env/bin/activate && cd /executorch && source
/executorch/examples/models/llama/install_requirements.sh
```

Build QNN backend for Executorch

```
./backends/qualcomm/scripts/build.sh --release

cmake -DPYTHON_EXECUTABLE=python \
    -DCMAKE_INSTALL_PREFIX=cmake-out \
    -DEXECUTORCH_ENABLE_LOGGING=1 \
    -DCMAKE_BUILD_TYPE=Release \
    -DEXECUTORCH_BUILD_EXTENSION_MODULE=ON \
    -DEXECUTORCH_BUILD_EXTENSION_DATA_LOADER=ON \
    -DEXECUTORCH_BUILD_EXTENSION_TENSOR=ON \
    -DEXECUTORCH_BUILD_QNN=ON \
    -DQNN_SDK_ROOT=${QNN_SDK_ROOT} \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -Bcmake-out .
```

```
cmake --build cmake-out -j16 --target install --config Release
cp -fv /executorch/cmake-out/backends/qualcomm/Py*
"/executorch/backends/qualcomm/python"
```

(Optional) Build Llama Runner for QNN backend

```
cmake -DPYTHON_EXECUTABLE=python \
    -DCMAKE_INSTALL_PREFIX=cmake-out \
    -DCMAKE_BUILD_TYPE=Release \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -DEXECUTORCH_BUILD_EXTENSION_TENSOR=ON \
    -DEXECUTORCH_BUILD_QNN=ON \
    -Bcmake-out/examples/models/llama \
    examples/models/llama

cmake --build cmake-out/examples/models/llama -j16 --config Release
```

Export

1. Downloads Checkpoint & Tokenizer

   ```
   wget
   "https://huggingface.co/karpathy/tinyllamas/resolve/main/stories110M.pt"
   wget
   "https://raw.githubusercontent.com/karpathy/llama2.c/master/tokenizer.model"
   ```

2. Edit params.json

   ```
   echo '{"dim": 768, "multiple_of": 32, "n_heads": 12, "n_layers": 12,
   "norm_eps": 1e-05, "vocab_size": 32000}' > params.json
   ```

3. QNN Export Command 3-1. Simple Export Command(No-Quantize)

   ```
   python -m examples.models.llama.export_llama -kv --disable_dynamic_shape --
   qnn -c stories110M.pt -p params.json
   ```

   3-2. 8A8W Quantization Export Command

```
python -m examples.models.llama.export_llama -kv --disable_dynamic_shape --
qnn --pt2e_quantize qnn_16a16w -d fp16 -c stories110M.pt -p params.json
```

4. Convert Tokenizer

```
python -m extenstion.llm.tokenizer.tokenizer -t tokenizer.model -o
tokenizer.bin
```

Run on your Android Device

Set Android NDK

```
export ANDROID_NDK=$ANDROID_NDK_ROOT
```

Build Executorch library for Android

```
cmake -DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake \
    -DANDROID_ABI=arm64-v8a \
    -DANDROID_PLATFORM=android-23 \
    -DCMAKE_INSTALL_PREFIX=cmake-out-android \
    -DCMAKE_BUILD_TYPE=Release \
    -DEXECUTORCH_BUILD_EXTENSION_DATA_LOADER=ON \
    -DEXECUTORCH_BUILD_EXTENSION_MODULE=ON \
    -DEXECUTORCH_BUILD_EXTENSION_TENSOR=ON \
    -DEXECUTORCH_ENABLE_LOGGING=1 \
    -DPYTHON_EXECUTABLE=python \
    -DQNN_SDK_ROOT=${QNN_SDK_ROOT} \
    -DEXECUTORCH_BUILD_QNN=ON \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -Bcmake-out-android .

cmake --build cmake-out-android -j16 --target install --config Release
```

Build Llama Runner for Android

```
cmake  -DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake \
    -DANDROID_ABI=arm64-v8a \
    -DANDROID_PLATFORM=android-23 \
    -DCMAKE_INSTALL_PREFIX=cmake-out-android \
    -DCMAKE_BUILD_TYPE=Release \
    -DPYTHON_EXECUTABLE=python \
    -DEXECUTORCH_BUILD_QNN=ON \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -Bcmake-out-android/examples/models/llama \
    examples/models/llama

cmake --build cmake-out-android/examples/models/llama -j16 --config Release
```

Upload Exported Model, Tokenizer and llama runner binary files

```
adb shell mkdir -p /data/local/tmp/llama/
adb push llama2.pte /data/local/tmp/llama/
adb push tokenizer.bin /data/local/tmp/llama/
adb push cmake-out-android/examples/models/llama/llama_main /data/local/tmp/llama/
```

Upload QNN backend .so files

```
adb push ${QNN_SDK_ROOT}/lib/aarch64-android/libQnnHtp.so /data/local/tmp/llama/
adb push ${QNN_SDK_ROOT}/lib/aarch64-android/libQnnSystem.so
/data/local/tmp/llama/
adb push ${QNN_SDK_ROOT}/lib/aarch64-android/libQnnHtpV69Stub.so
/data/local/tmp/llama/
adb push ${QNN_SDK_ROOT}/lib/aarch64-android/libQnnHtpV73Stub.so
/data/local/tmp/llama/
adb push ${QNN_SDK_ROOT}/lib/aarch64-android/libQnnHtpV75Stub.so
/data/local/tmp/llama/
adb push ${QNN_SDK_ROOT}/lib/hexagon-v69/unsigned/libQnnHtpV69Skel.so
/data/local/tmp/llama/
adb push ${QNN_SDK_ROOT}/lib/hexagon-v73/unsigned/libQnnHtpV73Skel.so
/data/local/tmp/llama/
adb push ${QNN_SDK_ROOT}/lib/hexagon-v75/unsigned/libQnnHtpV75Skel.so
/data/local/tmp/llama/
```

Connecting with Android and Hexagon dynamic linkers through Library PATH

```
export LD_LIBRARY_PATH=/data/local/tmp/llama/ && export
ADSP_LIBRARY_PATH=/data/local/tmp/llama/
```

Now, You can try to run the model on your device

```
adb shell
cd /data/local/tmp/llama
```

```
chmod 777 llama_main
./llama_main --model_path llama2.pte --tokenizer_path tokenizer.bin --prompt Once
upon a time,
```

Build Android Demo App

Build AAR Library

1. Setup Environments `export ANDROID_ABI=arm64-v8a`
2. Build Android JAVA Extenstion Code

   ```
   pushd extension/android
   ./gradlew build
   popd
   ```

3. Setup JNI Library

   ```
   pushd examples/demo-apps/android/LlamaDemo
   ./gradlew :app:setupQnn
   popd
   ```

And then, **executorch.aar** file will be generated in a newly created folder in **examples/demo-apps/android/LlamaDemo/app/libs** directory.

4. Run the Android Demo App

   ```
   export ANDROID_HOME=/opt/android-sdk
   pushd examples/demo-apps/android/LlamaDemo
   ./gradlew :app:installDebug
   popd
   ```